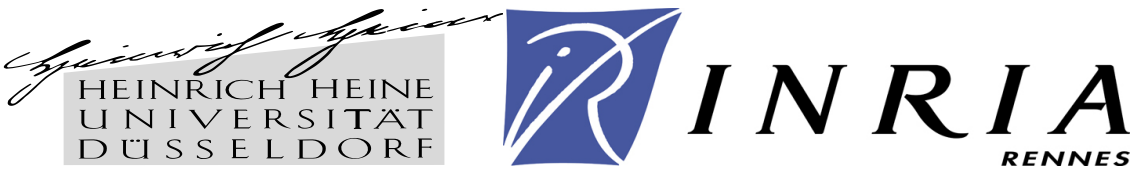
Unabhängiges Checkpointing in einer heterogenen Grid-Umgebung

Eugen Feller

Masterarbeit Präsentation

Gutachter: Prof. Dr. Michael Schöttner
Dr. Christine Morin
Betreuer: Dipl.-Inf. John Mehnert-Spahn

November 25, 2009



Übersicht

Übersicht

- Checkpointing und Recovery
  - Periodische Sicherung des Anwendungszustandes auf einem persistenten Speicher
  - Im Fehlerfall wird der letzte, fehlerfreie Zustand wiederhergestellt
- Zwei mögliche Checkpoint-Strategien
  - Koordiniert
  - Unkoordiniert (mit oder ohne Nachrichtenaufzeichnung)
- Ziel dieser Arbeit
  - Entwurf und Implementierung des unabhängigen Checkpointings ohne Nachrichtenaufzeichnung im Kontext von XtreemOS
  - Leistungsbewertung der Implementierung

Einführung in Grids

Grundlagen

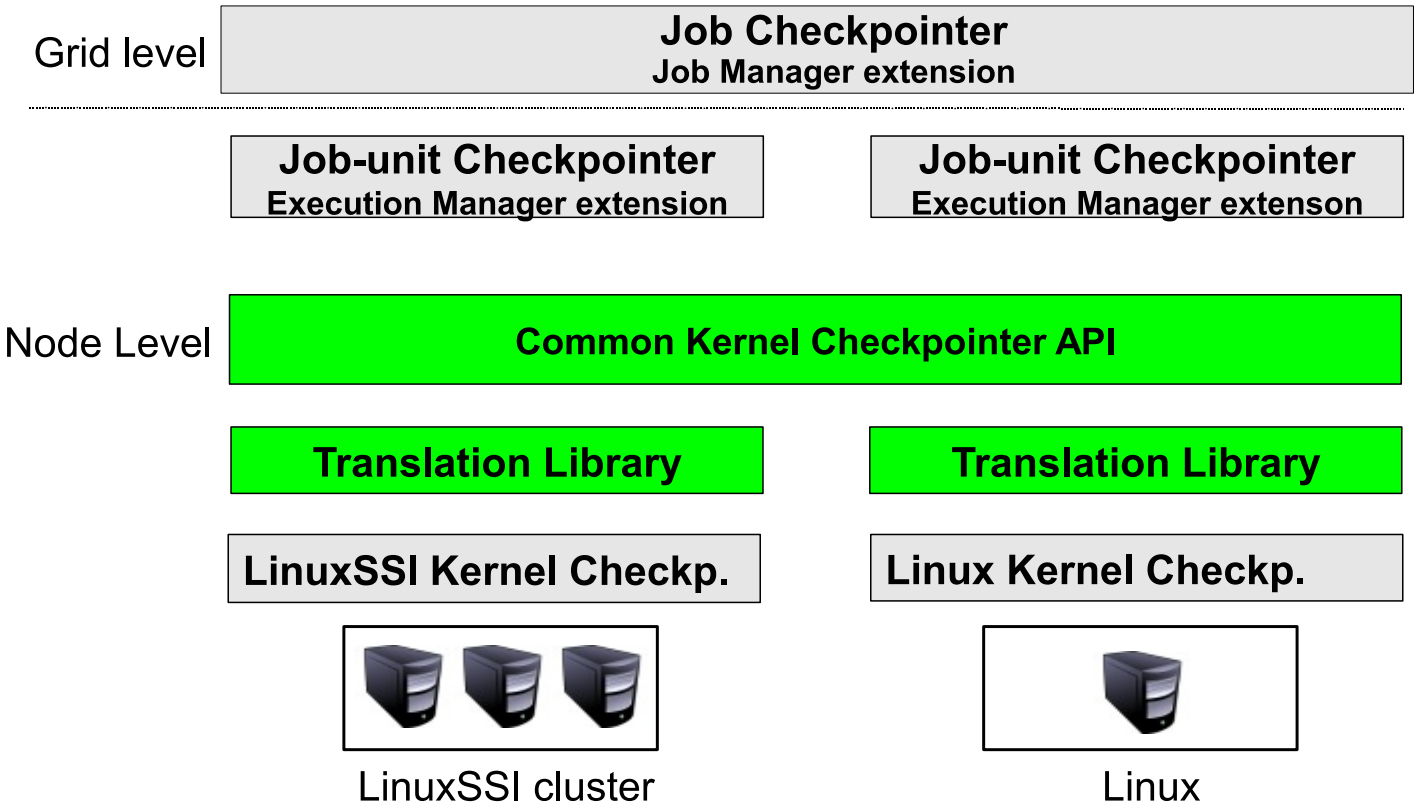
- Eigenschaften von Grids:
  - Offene Schnittstellen, Lose Kopplung, Heterogenität, etc.
- Unterschiedliche Grid-Technologien:
  - Middleware
  - Verlagerung der wichtigen Gridfunktionalitäten in das Betriebssystem
- XtreemOS: Linux-basiertes Grid-Betriebssystem
  - Erweitert um VO-Verwaltung und andere Grid-Dienste (z.B. Checkpointing)
  - Anforderung der Ressourcen mittels der POSIX und SAGA-Schnittstellen

XtreemOS Grid-Checkpointer

Grundlagen

- Verteilte Anwendungen in einer Grid-Umgebung
  - Mehrere unterschiedliche Grid-Knoten (z.B. PC oder Cluster)
  - Unterschiedliche Checkpointer (z.B. BLCR, LinuxSSI, etc.)
- XtreemOS Grid-Checkpointer (XtreemGCP)
  - Verteilter Dienst
  - Fehlertoleranz auf der Grid-Ebene
  - Implementiert die Checkpoint-Strategien (Koordiniert, Unkoordiniert, etc.)
  - Verwaltet die Checkpointer der Grid-Knoten
- “Common Kernel Checkpointer API”
  - Transparente Ansteuerung der darunter liegenden Checkpointer

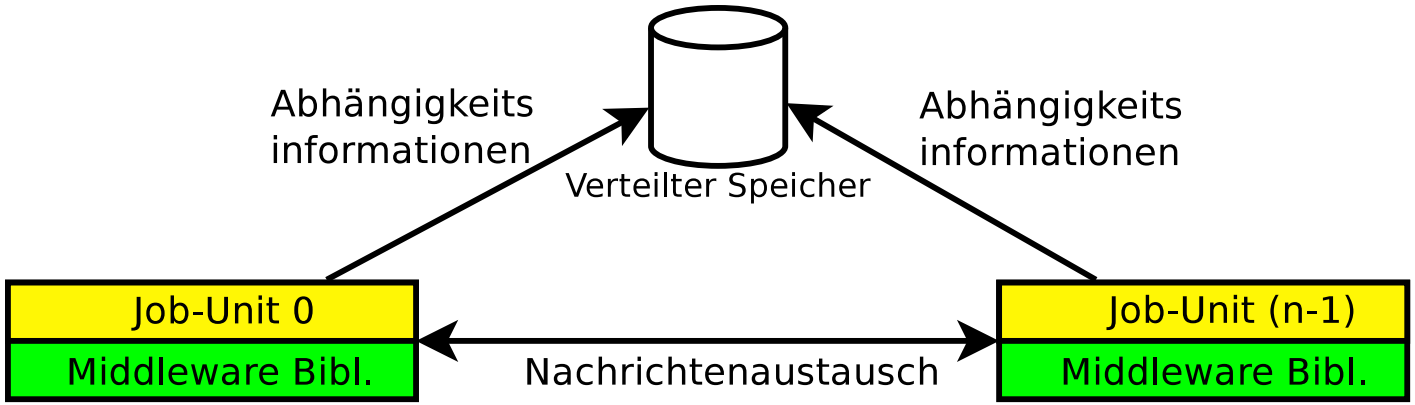
Aufbau


Grid level
Job Checkpointer
Job Manager extension
Job-unit Checkpointer
Execution Manager extension
Job-unit Checkpointer
Execution Manager extenson
Node Level
Common Kernel Checkpointer API
Translation Library
Translation Library
LinuxSSI Kernel Checkp.
Linux Kernel Checkp.
LinuxSSI cluster
Linux

Aufzeichnung der Abhängigkeiten

Inhalt der Abhängigkeitsinformationen:

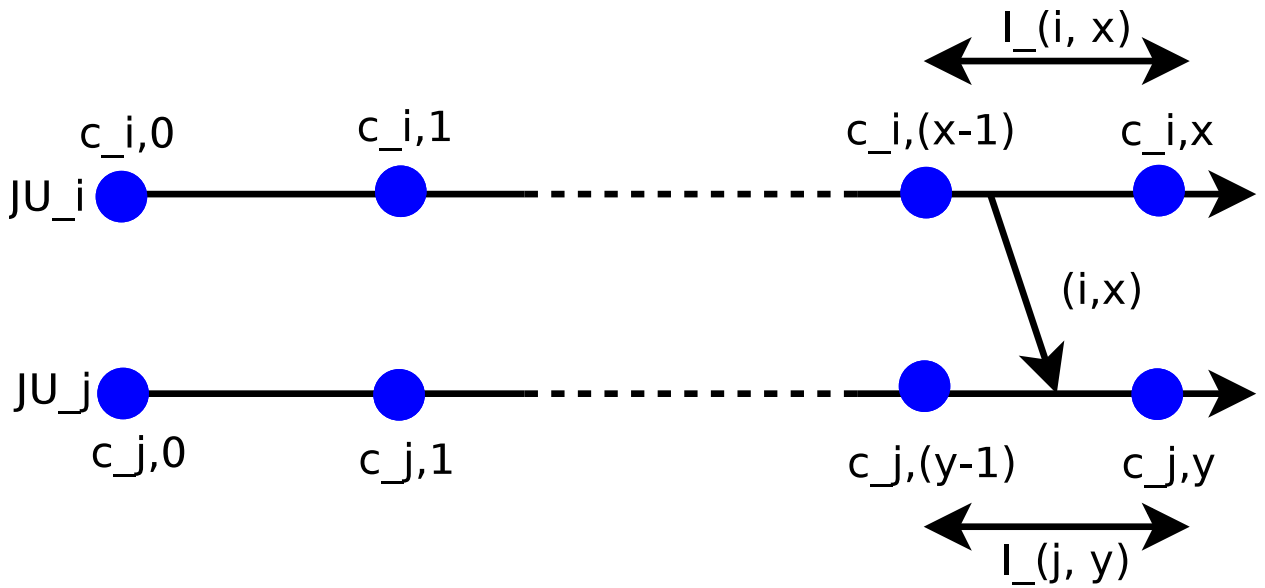
- Job-Unit ID
- Checkpoint-Index



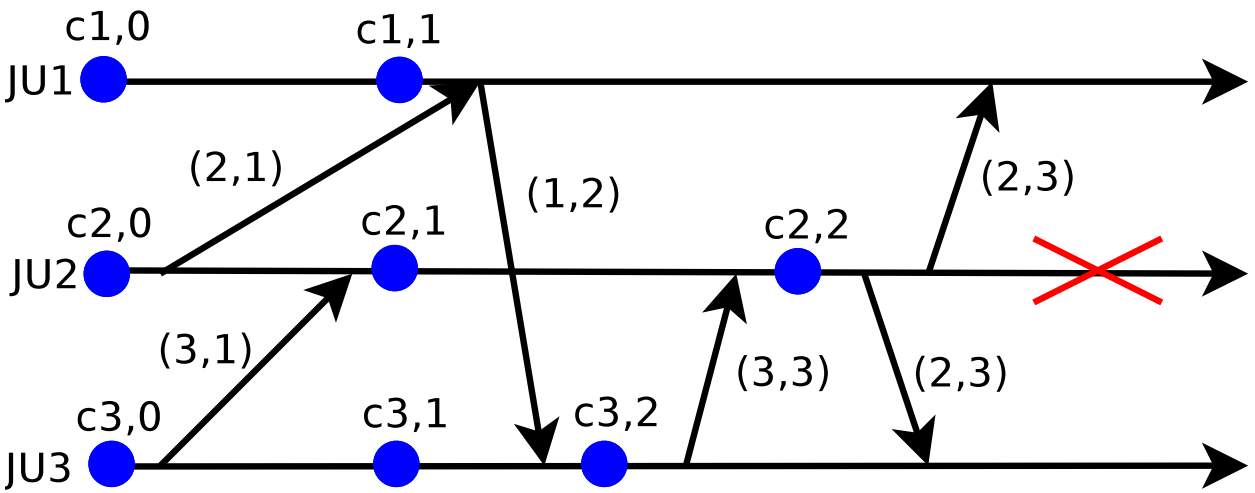
Checkpoint-Index und Checkpoint-Intervall

Definitionen:

- $c_{i,x} = x - te$ checkpoint $(x \geq 0)$ der Job-Unit JU_i $(0 \leq i \leq N-1)$
- $I_{i,x}$ = Checkpoint-Intervall zwischen zwei Sicherungen $c_{i,x-1}$ und $c_{i,x}$



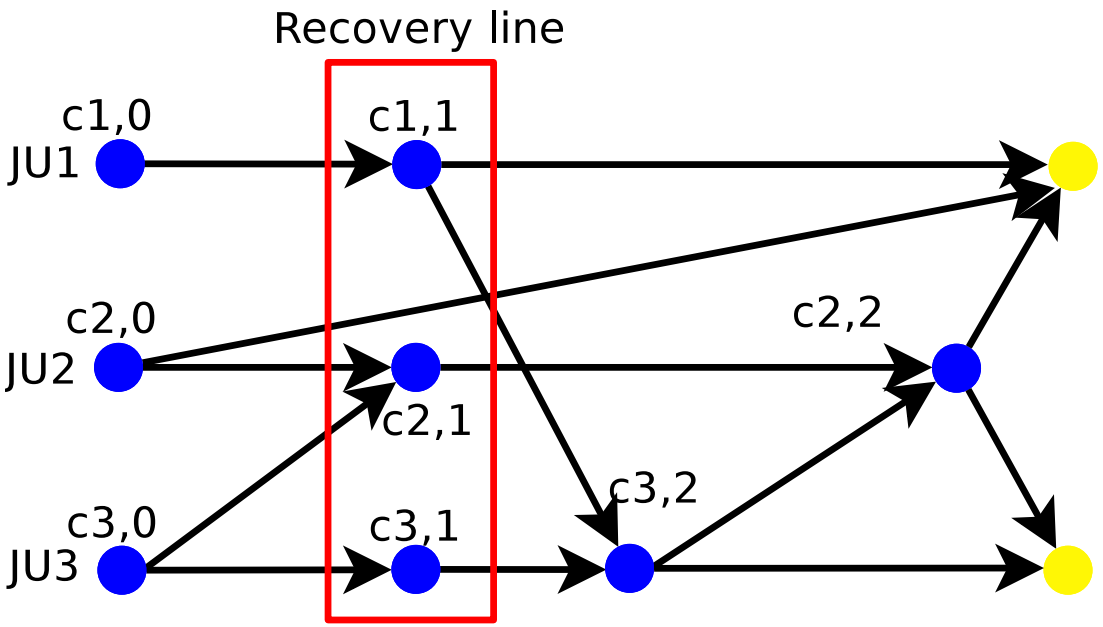
Verteilte Anwendung mit drei Job-Units



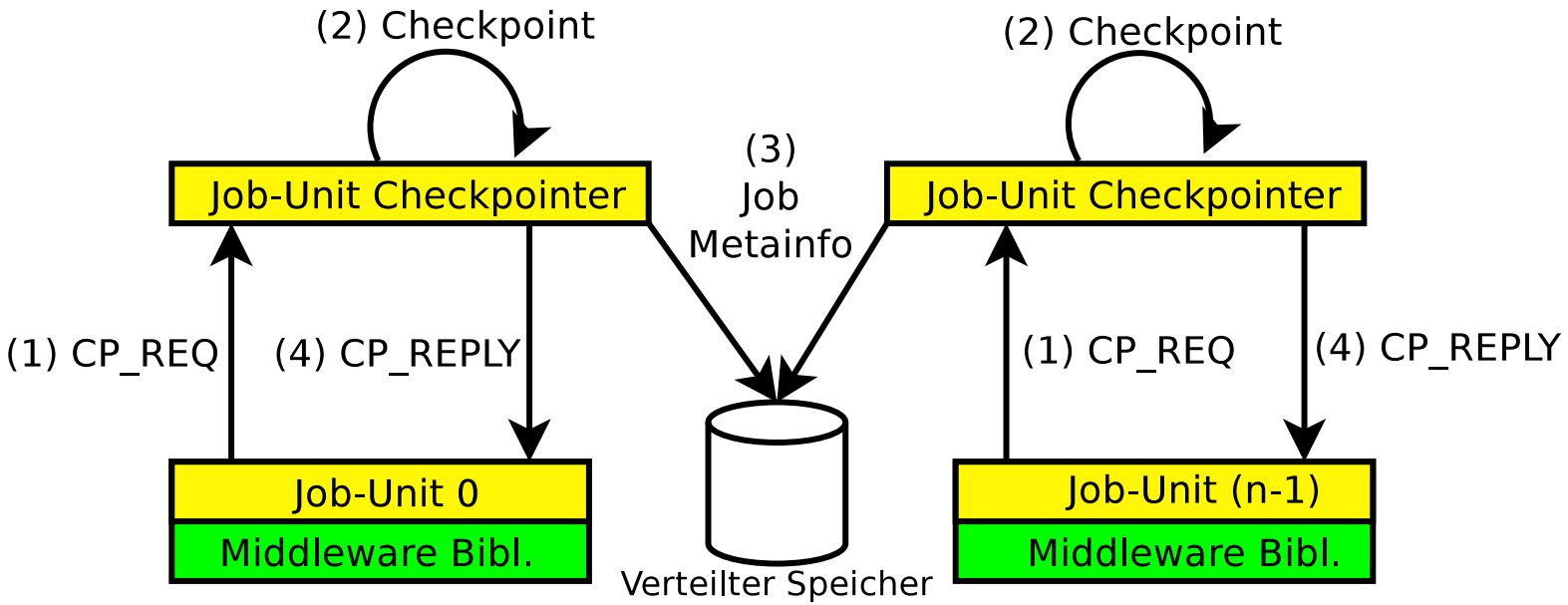
Checkpoint-Graph und Rollback-Propagation Algorithmus

Gerichtete Kante von $c_{i,x-1}$ aus nach $c_{j,y}$ genau dann wenn:

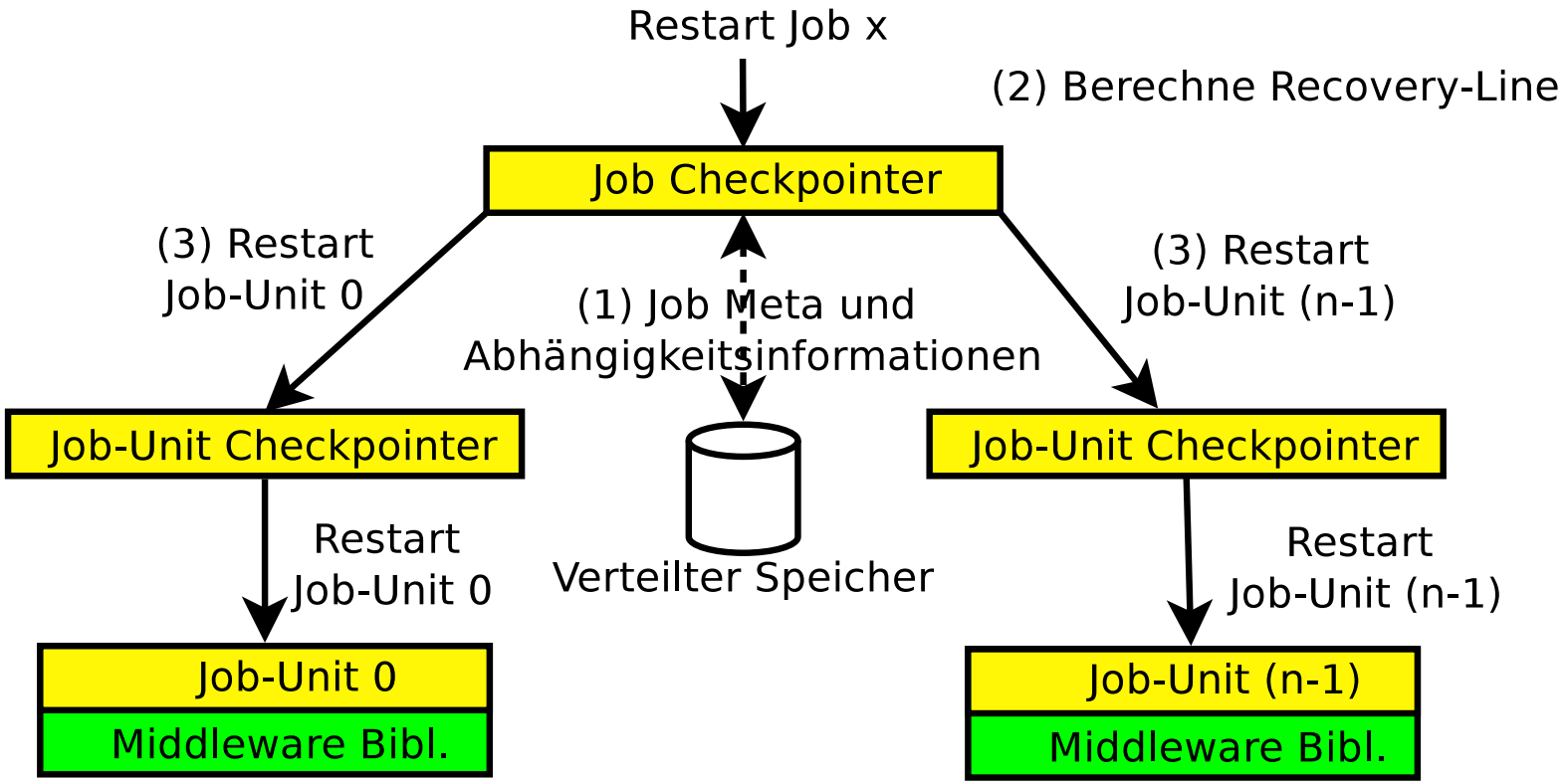
- $i \neq j$ und eine Nachricht wurde von $I_{i,x}$ aus geschickt und in $I_{j,y}$ empfangen oder
- $i = j$ und $y = x + 1$



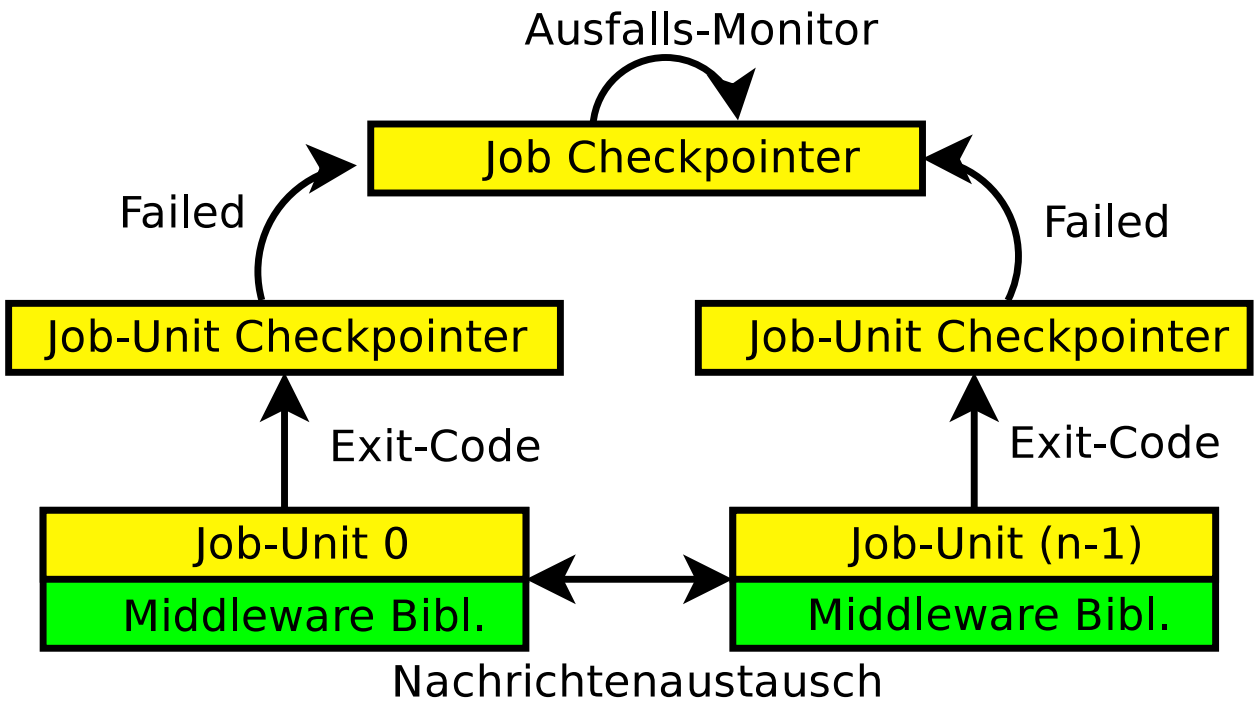
Durchführung einer unabhängigen Sicherung



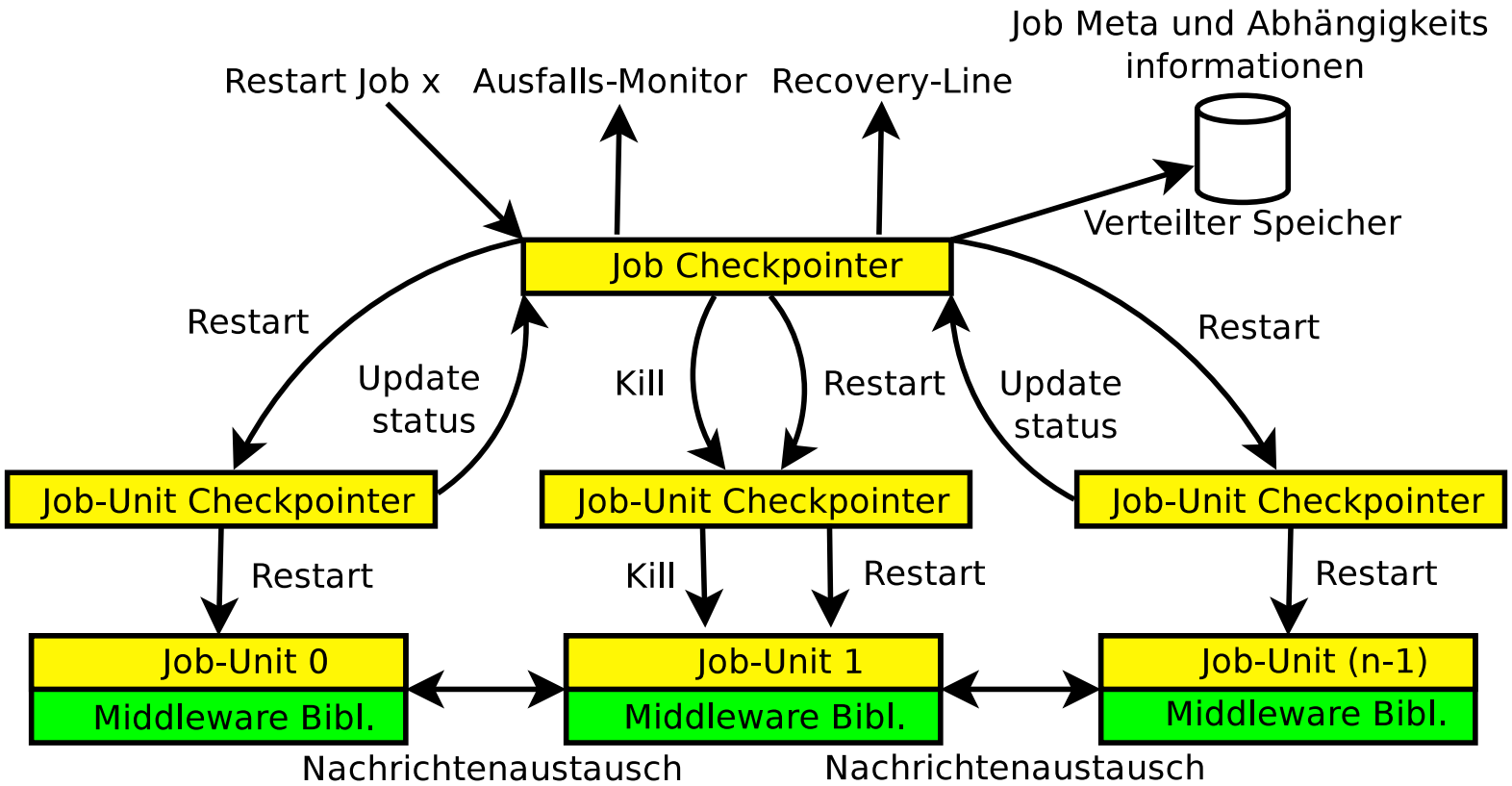
Ausfall der gesamten Anwendung



Teilausfall der Anwendung - Ausfallmonitor



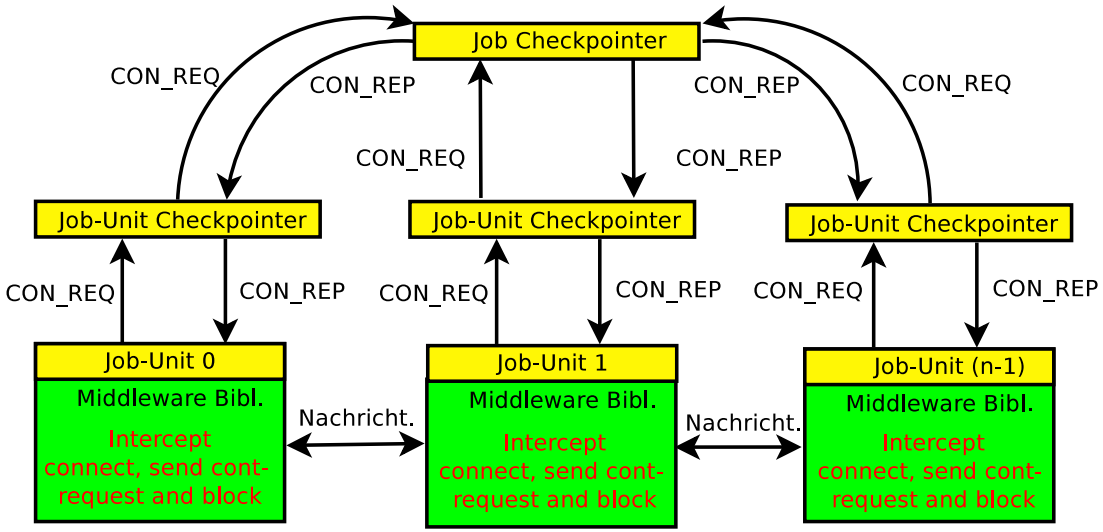
Teilausfall der Anwendung - Wiederherstellung



Vorgehensweisen bei der Wiederherstellung

Fragestellung: Welche Job-Unit muss zuerst wiederhergestellt werden? Zwei mögliche Vorgehensweisen:

- Zunächst die Server, dann die Clients
- Keine explizite Unterscheidung

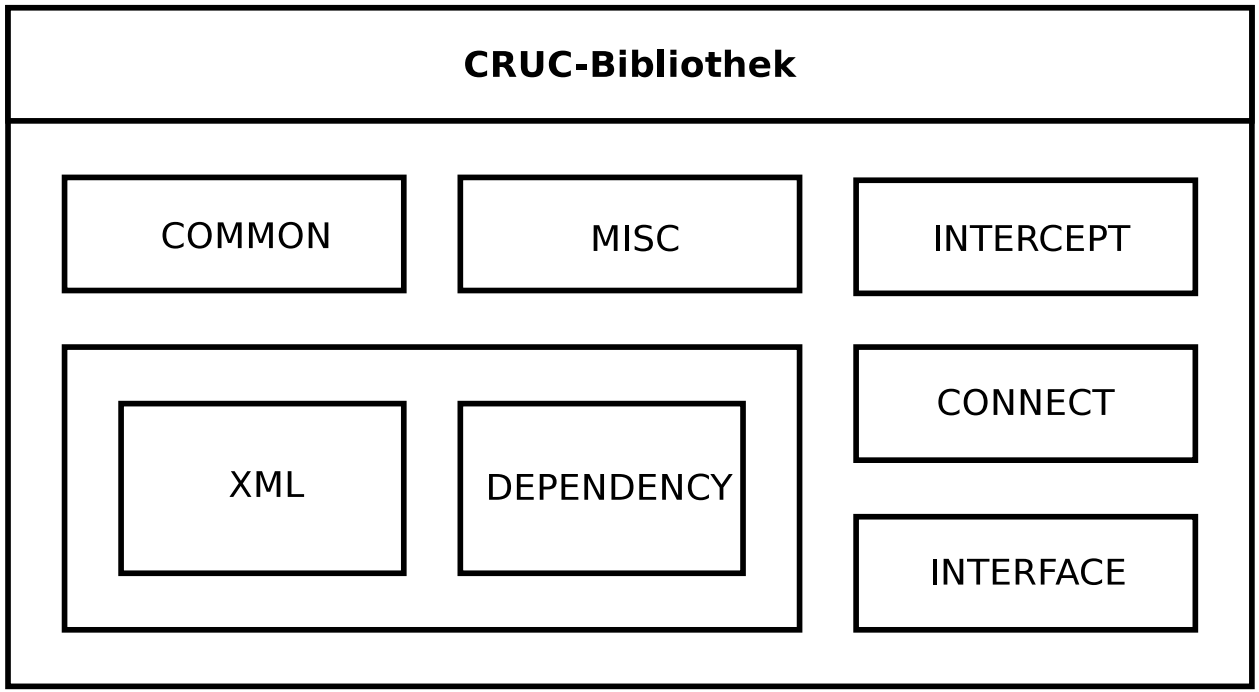


Abfangen von Bibliotheks- oder Systemaufrufen

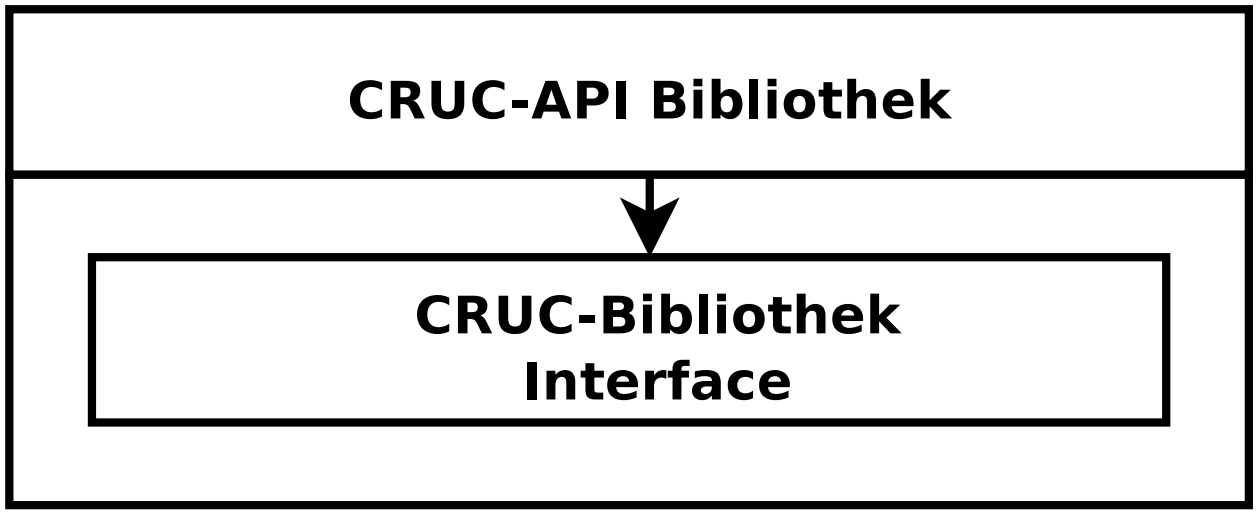
Abfangen von Bibliotheks- oder Systemaufrufen

- Hintergrund
  - Anhängen und extrahieren der Abhängigkeitsinformationen
- Zwei Vorgehensweisen
  - Bibliotheksbasiert
    - Leicht implementierbar und effizient
    - Software-Interrupts werden nicht beachtet
  - Kernel-Ebene
    - Ersetzten der Systemaufrufe mit eigenen Routinen
    - Änderungen betreffen das gesamte System
    - Auferlegen von Zugriffsrechten
    - Eingeschränkte Einsatzfähigkeit

Aufbau der CRUC-Bibliothek



Dynamisches Laden der CRUC-Bibliothek



Erweiterungen des Grid-Checkpointers

Erweiterungen des Grid-Checkpointer

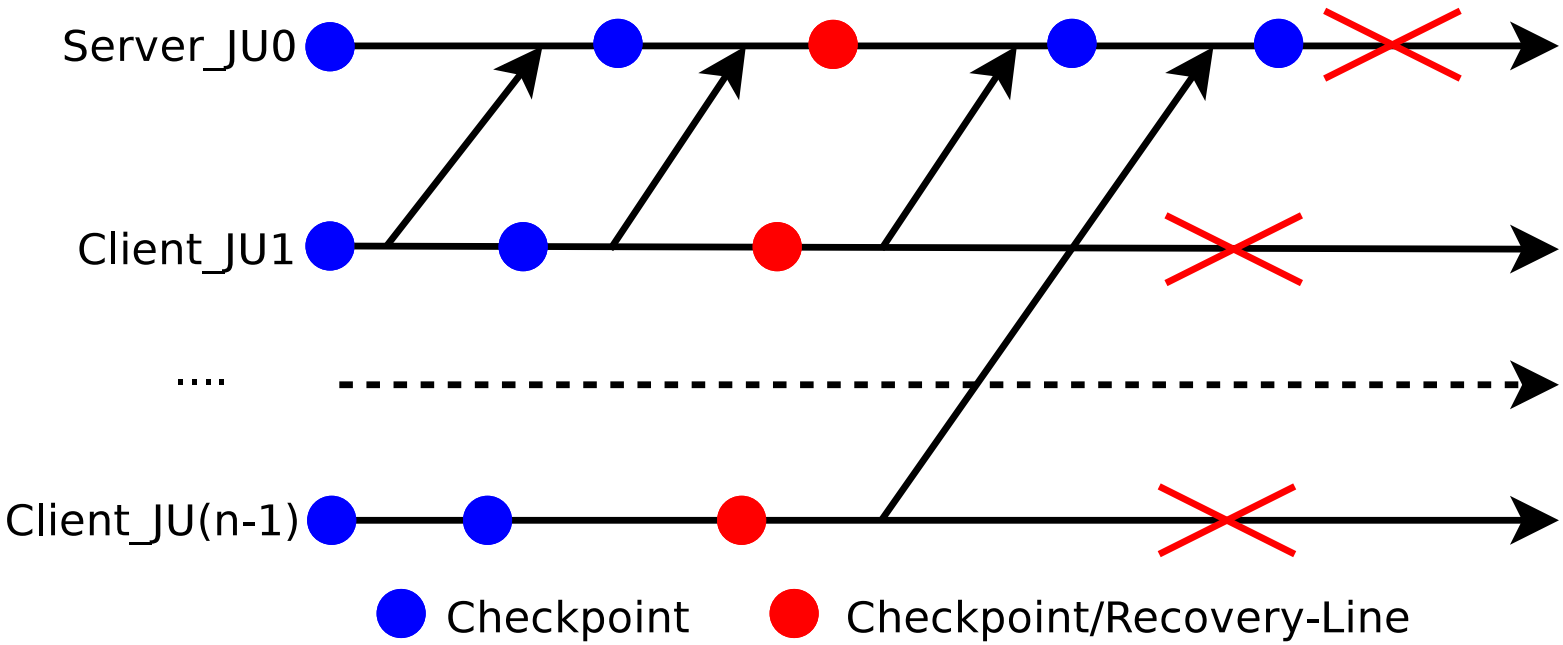
- Job-Unit Checkpointer
  - Nimmt Anfragen der CRUC-Bibliothek entgegen
  - Wählt einen passenden Checkpointer aus und erstellt eine Sicherung
  - Aktualisiert die Job-Meta Informationen
- Execution-Manager
  - Erkennt und leitet Anwendungsausfälle an den Job-Checkpointer weiter
- Job-Checkpointer
  - Liest die Abhängigkeitsinformationen aus
  - Implementiert den Rollback-Propagation Algorithmus
  - Stellt die Anwendung in einen konsistenten Zustand wieder her

Ausgangssituation

Ausgangssituation

- Heterogene Grid-Umgebung
  - Vier PC-Knoten (mit BLCR)
  - Ein LinuxSSI-Cluster (zwei Knoten)
- Modifizierte Version der AEM auf allen Knoten
- Client-Server Anwendung
  - Bestehend aus sechs Job-Units
  - Server auf einem der PC-Knoten
  - Clients verteilt auf die restlichen Knoten
- Messungen
  - Aufzeichnung von Abhängigkeiten
  - Zeit zur Berechnung einer Recovery-Line
  - Sicherungs- und Wiederherstellungszeit

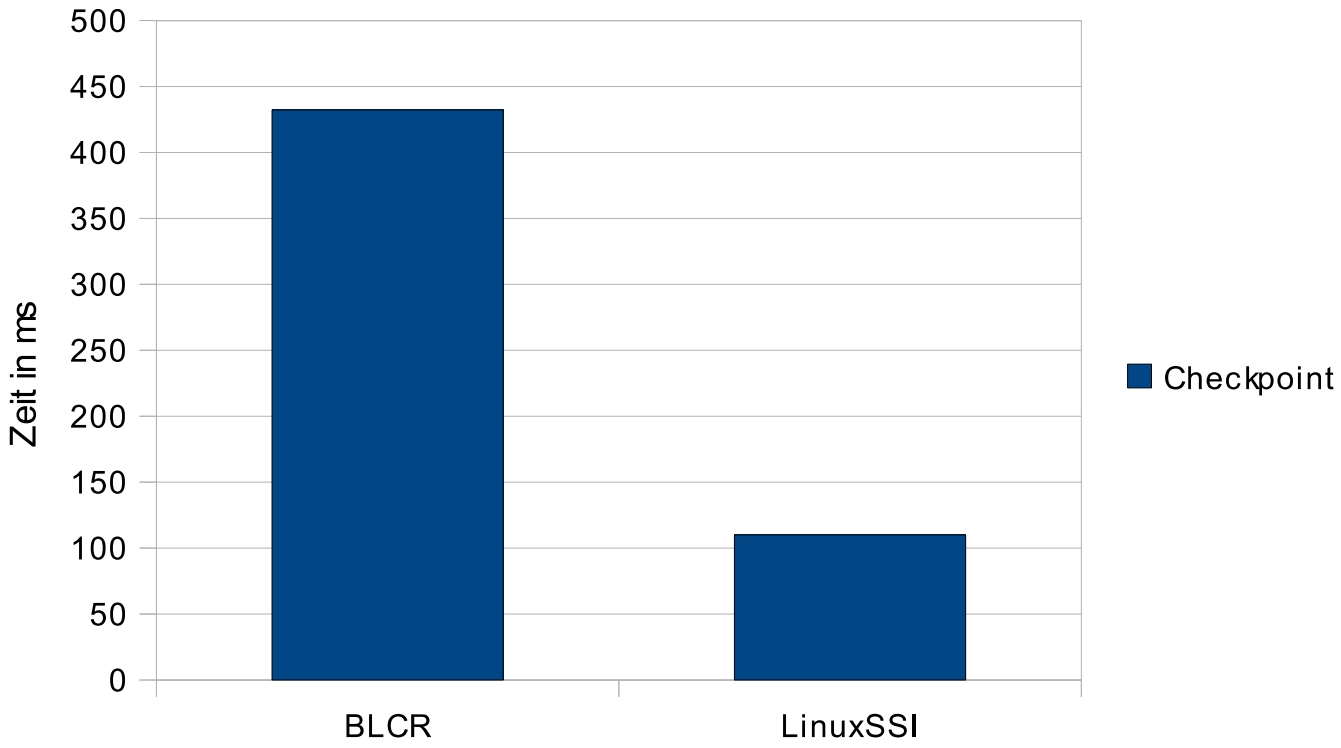
Zugriffsmuster der Anwendung



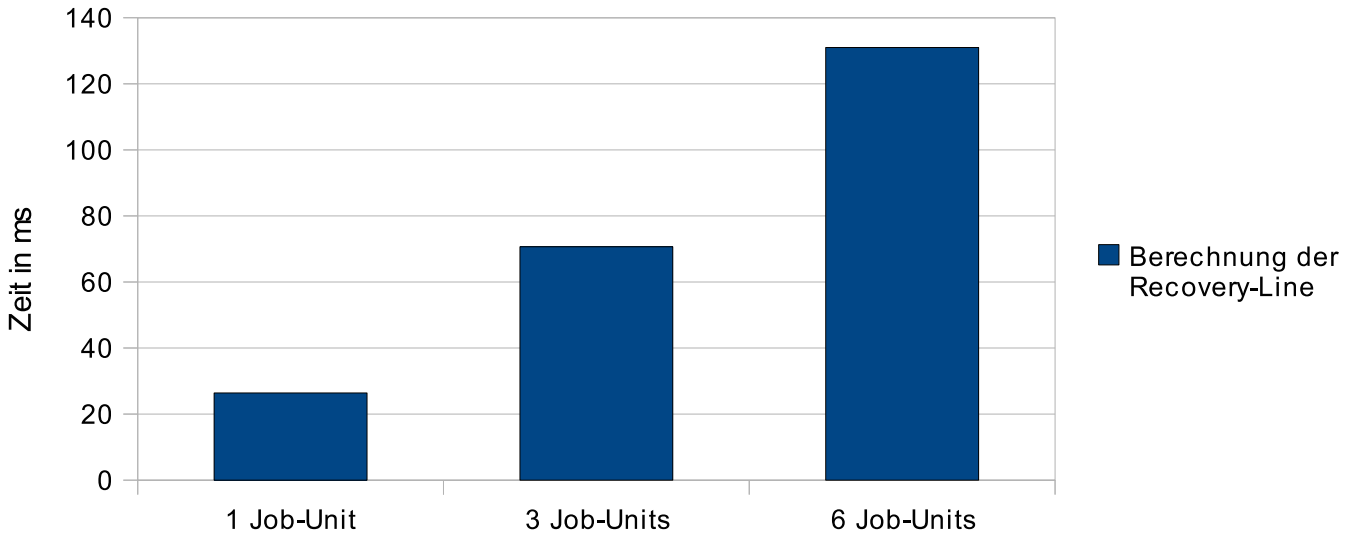
Verwaltung der Abhängigkeitsinformationen

- Native Sende- und Empfangsroutinen wurden abgefangen und modifiziert
  - Anhängen der Abhängigkeitsinformationen
- Fragestellung: Wie gross ist der Aufwand für diese Prozedur?
  - Unterschied nur marginal: $< 1\mu s$
- Kein Kopieren der Nachrichten zum Anhängen der Abhängigkeitsinformationen notwendig

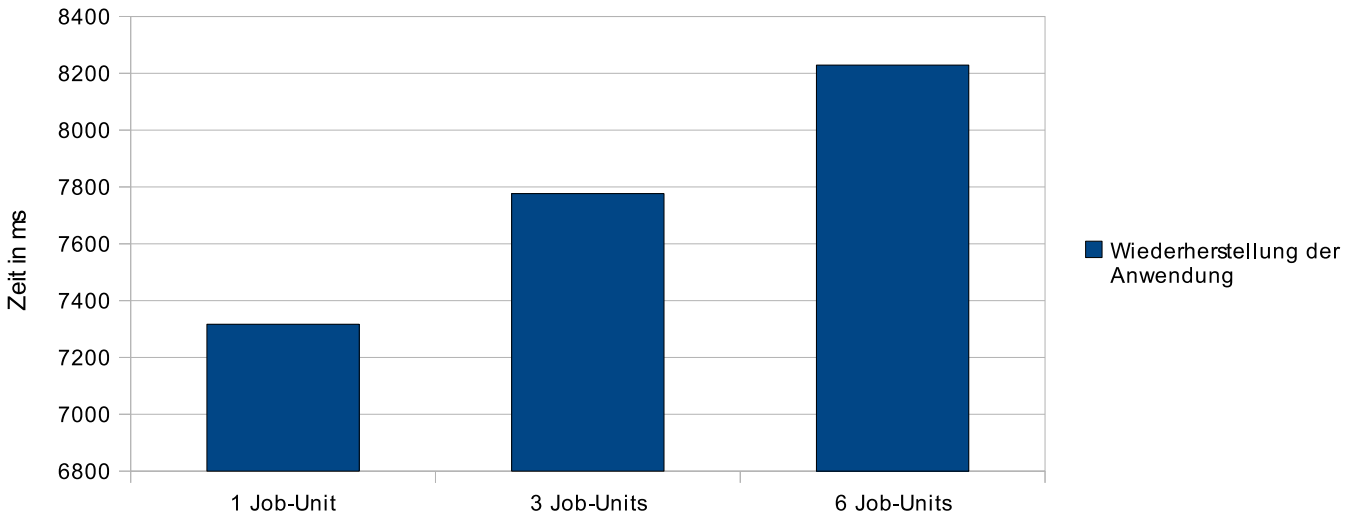
Durchschnittliche Dauer für die Erstellung einer Sicherung



Durchschnittliche Dauer für die Berechnung der Recovery-Line



Durchschnittliche Dauer für Wiederherstellung der Anwendung



Zusammenfassung

Zusammenfassung

- Konzept wurde vollständig in die XtreemOS Grid-Checkpointer Komponente integriert
- Dedizierte Middleware-Bibliothek kümmert sich um die Aufzeichnung der Abhängigkeiten
- Erweiterungen des Grid-Checkpointers
  - Verarbeitet die Checkpoint-Anfragen und erstellt die Sicherungen
  - Kümmert sich um die Ausfallerkennung
  - Implementiert den Rollback-Propagation Algorithmus
  - Stellt die Anwendung im Fehlerfall wieder her
- Messungen innerhalb einer heterogenen Grid-Umgebung
  - Belegten die Tauglichkeit der Implementierung

Optimierungs- und Erweiterungsmöglichkeiten

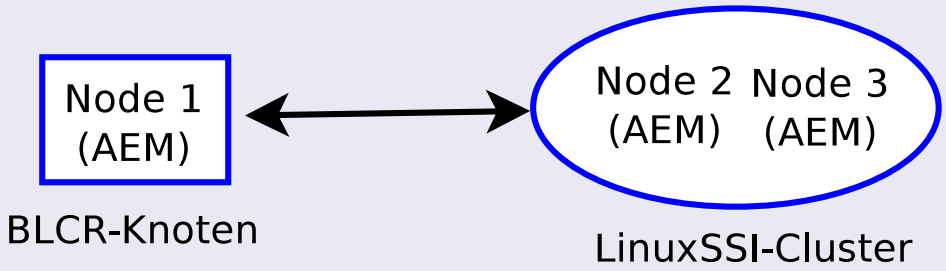
Überblick

- Speicherung der Abhängigkeitsinformationen neben den Job-Meta Informationen
- Wiederherstellung von Job-Units, die sowohl als Client wie auch als Server agieren
- Berechnung eines global konsistenten Zustandes ausgehend von einer vorgegeben Version
- Aufzeichnung der Nachrichtenübertragung (Vermeidung des Domino-Effekts)
- Integration eines Garbage-Collection Algorithmus
- Adaptives Umschalten der Checkpoint-Strategien

Live Demo

Aufbau der Demo-Umgebung

- Heterogene Grid-Umgebung



- Modifizierte Version der AEM auf allen Knoten
- Client-/Server Testanwendung
  - Server auf dem BLCR-Knoten
  - Clients auf dem LinuxSSI-Cluster
- AEM-Konsole: Erstellung einer Reservierung (drei Knoten), Ausführung der Anwendung

Vielen Dank für Ihre Aufmerksamkeit!